PATENT ABSTRACTS OF JAPAN

(11)Publication number : **08-016385**
(43)Date of publication of application : **19.01.1996**

(51)Int.Cl. **G06F 9/06**
G06F 15/00

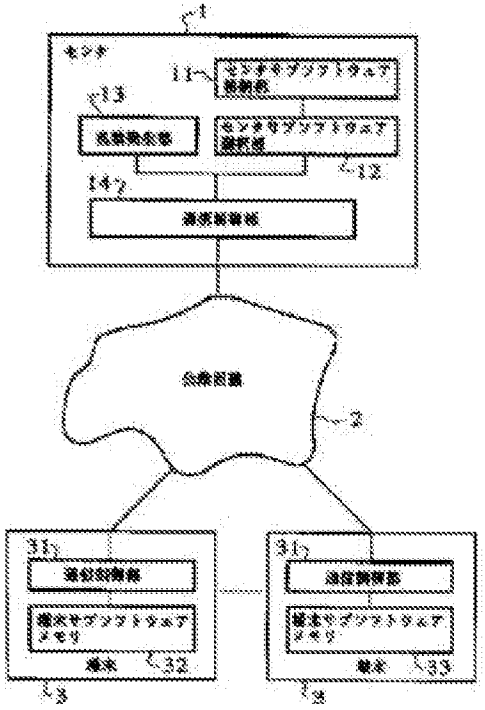
(21)Application number : **07-105177**
(22)Date of filing : **28.04.1995**

(71)Applicant : **NIPPON TELEGR & TELEPH CORP <NTT>**
(72)Inventor : **OKUYAMA HIRONOBU**
MORIYASU KENJI
KANAI ATSUSHI
MIYAKE NOBUHISA
TERAUCHI ATSUSHI

(30)Priority
Priority number : **06 92134** Priority date : **28.04.1994** Priority country : **JP**

(54) PROTECTING METHOD FOR SOFTWARE ANALYSIS

(57)Abstract:
PURPOSE: To provide the protecting method for software analysis with which a software can be surely prevented from being illegally used by making the analysis difficult without complicating or enciphering the software itself.
CONSTITUTION: An application software is divided at both a center 1 and a terminal 3 and respectively held in a center sub-software storage part 11 and a terminal sub-software memory 32, and the lacked part of the application software to be down loaded to the terminal 3 is suitably selected by a random number generating part 13 and a center sub-software selecting part 12 at the center 1 and down loaded to the terminal 3. Thus, the arrangement of the application software on the memory at the terminal 3 is set so as to be various each time its is used, and all the application softwares are not simultaneously extended on the memory.



CLAIMS

[Claim(s)]

[Claim 1] In a system which comprised two or more terminals connected to a center and this center, It is a software analysis protective method protected so that software used at a terminal may not be analyzed at a terminal, Distribute a software portion which divided software used at a terminal to a terminal and a center, and it is held, Whenever it uses said software in a terminal, a software portion held from a terminal at a center is required, By loading a software portion which transmitted to a terminal a software portion held at a center according to a demand from a terminal, and was transmitted from a center to a missing part in a software portion held at a terminal on a memory, A software analysis protective method which consists of what operation of said software is enabled on a terminal, and said software developed on said memory is operated for on a terminal changing an arrangement pattern developed on said memory of said software whenever said software is used at a terminal.

[Claim 2] In the software analysis protective method according to claim 1, a software portion held at a center contains a group of two or more software portions with a different function, A thing, wherein a center changes a function of said software to make it operate at a terminal by choosing a group of one software portion which transmits to a terminal by judgment of a demand or a center from a terminal.

[Claim 3] In the software analysis protective method according to claim 1, a software portion held at a center contains a group of two or more software portions with the same function, A thing whenever said software is used at a terminal an arrangement pattern developed on said memory of said software by choosing a group of one software portion which transmits to a terminal based on a random number, wherein a center changes.

[Claim 4] In the software analysis protective method according to claim 1, a software portion held at a center contains a group of two or more software portions, A thing when a center chooses a group of one software portion which transmits to a terminal, and a group of this software portion transmitted from a center is loaded to a corresponding missing part on said memory, wherein operation of said software is selectively attained at least on a terminal.

[Claim 5] A thing, wherein a center transmits a group of two or more of said software portions one by one in the software analysis protective method according to claim 1 according to a demand from a terminal accompanying advance of operation of said software.

[Claim 6] A thing, wherein a software portion loaded to a missing part is cleared from said memory after an end of use of said software in the software analysis protective method according to claim 1.

[Claim 7] A thing, wherein a software portion loaded to a missing part is cleared from said memory one by one working [said software] in the software analysis protective method according to claim 1 as it becomes unnecessary for operation after said software.

[Claim 8] A thing, wherein a software portion loaded to a missing part is cleared from said memory in the software analysis protective method according to claim 1 by command provided in a software portion held at a terminal.

[Claim 9] A thing, wherein a software portion loaded to a missing part is cleared from said memory in the software analysis protective method according to claim 1 by command provided in a software portion which was held at a center and loaded to a missing part.

[Claim 10] The maximum use count of each of this software portion of known [portion / each / which was held in the software analysis protective method according to claim 1 at a center / software] , A thing when a count of said number of times of actual use reaches [said maximum use count] this each software portion that has a count of the number of times of actual use of each of this software portion, and was loaded to a missing part, wherein it is cleared from said memory.

[Claim 11] A thing, wherein software portions held in the software analysis protective method according to claim 1 at a terminal are two or more discontinuous blocks in said software.

[Claim 12] Software portions held in the software analysis protective method according to claim 1 at a center are two or more discontinuous blocks in said software, a discontinuous block of a software portion held at a center -- this -- a thing transmitting in different turn from an order between discontinuous block comrades.

[Claim 13] It has a certain code which returns a certain fixed value which shows that each software portion held at a center is a software portion in which it was held at a center in the software analysis protective method according to claim 1 to a predetermined address in this each software portion, A thing when each missing part on said memory has another code which returns another fixed value it indicates it to be that it is a missing part to a predetermined address in this each missing part and said another fixed value is returned working [said software] on a terminal, wherein operation of said software is interrupted.

[Claim 14] An analysis protective method comprising:
A center.
It is a software analysis protective method protected so that software used at a terminal may not

be analyzed at a terminal in a system which comprised two or more terminals connected to this center, Distribute a software portion which divided software used at a terminal to a terminal and a center, and it is held, By loading a software portion which transmitted a software portion held at a center to a terminal, and was transmitted from a center to a missing part in a software portion held at a terminal on a memory, What operation of said software is enabled on a terminal, and said software developed on said memory is operated for on a terminal

[Claim 15] A thing whenever a software portion held at a center uses said software in a terminal in the software analysis protective method according to claim 14, wherein it is transmitted to a terminal from a center according to a demand from a terminal.

[Claim 16] A thing, wherein a software portion held at a center is divided and transmitted to two or more blocks in the software analysis protective method according to claim 14.

[Claim 17] A thing whenever said software is used at a terminal in the software analysis protective method according to claim 14, wherein an arrangement pattern developed on said memory of said software is changed.

[Claim 18] In the software analysis protective method according to claim 14, a software portion held at a center contains a group of two or more software portions, A thing when a center chooses a group of one software portion which transmits to a terminal, and a group of this software portion transmitted from a center is loaded to a corresponding missing part on said memory, wherein operation of said software is selectively attained at least on a terminal.

[Claim 19] A thing, wherein a software portion loaded to a missing part is cleared from said memory after an end of use of said software in the software analysis protective method according to claim 14.

[Claim 20] A thing, wherein a software portion loaded to a missing part is cleared from said memory one by one working [said software] in the software analysis protective method according to claim 14 as it becomes unnecessary for operation after said software.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]
[Industrial Application] This invention, So that software, such as an application program saved and executed at a terminal in the system which consists of a center and two or more terminals, may not be analyzed at a terminal. It is related with the software analysis protective method which protects and prevents the illegal use of software, such as an application program in a terminal.

[0002]
[Description of the Prior Art] Generally, to the software on a terminal, the illegal use (software analysis / alteration) by a malicious terminal user may occur. The illegal use of software begins from analyzing the software on a memory and a disk with tools, such as a debugger. In order to make this software analysis difficult, there is the method of complicating the software itself. However, while the effect of this method is a problem of a grade, it cannot apply this method to short software, and cannot be said as perfectness. Although the mechanism in which the method of enciphering software decrypts the software is needed, since the mechanism itself cannot be enciphered, there is a fault that raw software will become clear, by analyzing the output portion.

[0003]
[Problem to be solved by the invention] As mentioned above, in order to prevent the illegal use of the software on a terminal, can consider how to complicate the software itself and the method of enciphering, but. Since the mechanism in which there is a problem of being inapplicable in short software while the method of complicating is a problem of a grade, and the method of enciphering decodes software is needed and this mechanism itself cannot be enciphered, By analyzing the output portion, there is a problem that software will become clear and any method has a problem.

[0004] Without enciphering without complicating the software itself, the place which this invention was made in view of the above, and is made into the purpose makes the analysis difficult, and there is in providing the software analysis protective method which prevents the illegal use of software accurately.

[0005]
[Means for solving problem] In order to attain the above-mentioned purpose, this invention (Claim 1), In the system which comprised two or more terminals connected to the center and this center, It is a software analysis protective method protected so that the software used at a terminal may not be analyzed at a terminal, Distribute the software portion which divided the software used at a terminal to a terminal and a center, and it is held, Whenever it uses said software in a terminal, the software portion held from the terminal at the center is required, By loading the software portion which transmitted to the terminal the software portion held at the center according to the demand from a terminal, and was transmitted from the center to the missing part in the software portion held at the terminal on a memory, A software analysis protective method which consists of what operation of said software is enabled on a terminal, and said software developed on said memory is operated for on a terminal changing the arrangement pattern developed on said memory of said software whenever said software is used

at a terminal.

[0006] In the above-mentioned software analysis protective method this invention (Claim 2), A center including a group of two or more software portions in which a software portion held at a center had a different function by choosing a group of one software portion which transmits to a terminal by judgment of a demand or a center from a terminal, A function of said software to make it operate at a terminal is changed.

[0007] In the above-mentioned software analysis protective method this invention (Claim 3), A center including a group of two or more software portions with the function that a software portion held at a center is the same, by choosing a group of one software portion which transmits to a terminal based on a random number, An arrangement pattern developed on said memory of said software is changed whenever said software is used at a terminal.

[0008] In the above-mentioned software analysis protective method this invention (Claim 4), A software portion held at a center contains a group of two or more software portions, A center chooses a group of one software portion which transmits to a terminal, and when a group of this software portion transmitted from a center is loaded to a corresponding missing part on said memory, operation of said software is selectively attained at least on a terminal.

[0009] In the above-mentioned software analysis protective method, a center transmits this invention (Claim 5) one by one according to a demand from a terminal accompanying advance of operation of said software by a group of two or more of said software portions.

[0010] A software portion by which this invention (Claim 6) was loaded to a missing part in the above-mentioned software analysis protective method is cleared from said memory after an end of use of said software.

[0011] In the above-mentioned software analysis protective method, this invention (Claim 7) is cleared from said memory one by one working [said software] as it becomes unnecessary [a software portion loaded to a missing part] for operation after said software.

[0012] This invention (Claim 8) is cleared from said memory in the above-mentioned software analysis protective method by command provided in a software portion in which a software portion loaded to a missing part was held at a terminal.

[0013] This invention (Claim 9) is cleared from said memory in the above-mentioned software analysis protective method by command provided in a software portion which a software portion loaded to a missing part was held at a center, and was loaded to a missing part.

[0014] In the above-mentioned software analysis protective method this invention (Claim 10), The maximum use count of each of this software portion of known [portion / each / which was held at the center / software] , It has a count of the number of times of actual use of each of this software portion, and this each software portion loaded to the missing part is cleared from said memory, when the count of said number of times of actual use reaches said maximum use count.

[0015] This invention (Claim 11) is characterized by the software portions held at the terminal being two or more discontinuous blocks in said software in the above-mentioned software analysis protective method.

[0016] In the above-mentioned software analysis protective method this invention (Claim 12), the discontinuous block of the software portion which the software portions held at the center are two or more discontinuous blocks in said software, and was held at the center -- this -- it is transmitted in different turn from an order between discontinuous block comrades

[0017] In the above-mentioned software analysis protective method this invention (Claim 13), It has a certain code which returns a certain fixed value which shows that each software portion held at the center is a software portion in which it was held at the center to the predetermined address in this each software portion, When each missing part on said memory has another code which returns another fixed value it indicates it to be that it is a missing part to the predetermined address in this each missing part and said another fixed value is returned working [said software] on a terminal, operation of said software is interrupted.

[0018] An analysis protective method this invention is characterized by that comprises the following.
This invention (Claim 14) is a center.
It is a software analysis protective method protected so that software used at a terminal may not be analyzed at a terminal in a system which comprised two or more terminals connected to this center, Distribute a software portion which divided software used at a terminal to a terminal and a center, and it is held, By loading a software portion which transmitted a software portion held at a center to a terminal, and was transmitted from a center to a missing part in a software portion held at a terminal on a memory, What operation of said software is enabled on a terminal, and said software developed on said memory is operated for on a terminal

[0019] In the above-mentioned software analysis protective method, this invention (Claim 15) is transmitted to a terminal from a center according to a demand from a terminal, whenever a software portion held at a center uses said software in a terminal.

[0020] In the above-mentioned software analysis protective method, this invention (Claim 16) divides into two or more blocks a software portion held at a center, and is transmitted.

[0021] The arrangement pattern in which this invention (Claim 17) was developed on said memory of said software in the above-mentioned software analysis protective method is changed whenever said software is used at a terminal.

[0022] In the above-mentioned software analysis protective method this invention (Claim 18), The software portion held at the center contains the group of two or more software portions, A center chooses the group of one software portion which transmits to a terminal, and when the group of this software portion transmitted from the center is loaded to the corresponding missing part on said memory, operation of said software is selectively attained at least on a terminal.

[0023] The software portion by which this invention (Claim 19) was loaded to the missing part in the above-mentioned software analysis protective method is cleared from said memory after the end of use of said software.

[0024] In the above-mentioned software analysis protective method, this invention (Claim 20) is cleared from said memory one by one working [said software] as it becomes unnecessary [the software portion loaded to the missing part] for the operation after said software.

[0025]
[Function] In the software analysis protective method of this invention, it transmits to a terminal and the portion which divides and holds the software used at a terminal in both a terminal and the center, and holds the center at the time of software using is used. It is made to differ whenever it uses arrangement of the software used at the terminal on a memory at this time, Or the function of software is changed by a user's demand or judgment of a center, About what was lost in the portion transmitted from the center among the software used at a terminal, the opportunity for a software image to be simultaneously developed on a memory by clearing one by one is made into the minimum. Furthermore, the portion to which after the end of use of software was always transmitted from the center is cleared.

[0026]
[Working example] Hereafter, the embodiment of this invention is described using Drawings.

[0027] Drawing 1 is a figure showing the composition of the system which enforces the software analysis protective method concerning one embodiment of this invention. This system comprises two or more terminals 3 connected with the center 1 and this center 1 via the public line 2.

[0028] Software, such as an application program used at each terminal 3, It is divided and saved in the terminal 3 and the center 1, and The inside of this software that is divided, is saved and is used at the terminal 3, A part of software which is saved to the terminal 3 and used at the terminal 3 will be called terminal subsoftware, and a part of software which is saved in the center 1 and used at the terminal 3 will be called center subsoftware. Here, center subsoftware is software which complements terminal subsoftware.

[0029] Here, it is aimed at all the software except the center 1 and the main part of the protocol control software which controls the data transfer between the terminals 3 for the software used at the terminal 3 mentioned above. For example, the data etc. which are used for the various application programs used at the terminal 3 or the program of the operation system level except the main part of transfer control, a database, etc. are meant, and these all serve as an applied object of this invention.

[0030] The system which consists of the center 1 to which this invention may be applied, and the terminal 3 may be a system which the case of a personal-computer-communications network also has, for example, and delivers application software from the center 1 side.

[0031] Especially this invention is suitable for a software circulation system which delivers to the terminal 3 for whenever [of use of an application software system / every] , and collects a

usage fee.

[0032] As shown in drawing 1, The center 1 via the center subsoftware selecting part 12, the random number generation part 13, and the public line 2 which choose the center subsoftware stored in the center subsoftware storage 11 which stores center subsoftware, and this center subsoftware storage 11. Having [and] the communication control part 14 for communicating with the terminal 3, the terminal 3 has the communication control part 31 for communicating with the center 1 via the public line 2, and the terminal subsoftware memory 32.

[0033] In an initial state, terminal subsoftware vacates two or more fields of a certain amount of size as two or more discontinuous blocks, and is developed on the memory 32. Although drawing 2 shows this situation, the area (below, a lack part is called) which gave the slash which starts with the address A000, B000, C000, D000, and E000 hits it. Thus, since terminal subsoftware has a lack part, even if it performs only this terminal subsoftware, it does not carry out operation normal as application software.

[0034] Terminal subsoftware containing a lack part on the terminal 3 is distributed in a form of a floppy disk or CD-ROM, is loaded from a drive of the terminal 3, or has a method distributed by a file transfer from the center 1, for example. Or it memorizes beforehand on a hard disk of the terminal 3, and may be arranged on a memory at the time of system starting and software starting. Data of a size of an address of a lack part on a memory and an area of each part is beforehand set to the center 1 side here.

[0035] Software included in a lack part of terminal subsoftware is held as center subsoftware in a form of a discontinuous block at the center 1.
By loading these to a lack part of the terminal 3, application software will become perfect. However, the application software may operate, if it compatible center subsoftware does not go into all lack parts.

[0036] In a method of forming a lack part for all the center subsoftware, there is a method of transmitting all the center subsoftware collectively from the center 1 side, and there is also a method of making it composition which operates even if it compatible center subsoftware does not go into all lack parts.

[0037] For example, if the software part of a basic function is stored on the memory of the terminal 3 at least when center subsoftware contains the software part of the basic function of application, and the software part of expanded function, There is a constitution method which becomes usable [the application software of the terminal 3] .

[0038] Or if a certain specific lack part (group) is buried, there is a constitution method which becomes usable [the application software of the terminal 3] .

[0039] Namely. [whether the lack part which begins from the address A000 like drawing 2, and the lack part which begins from the address C000 are buried with center subsoftware, and] Or it is the constitution method of operating in any [of whether the lack part which begins from the address B000, the lack part which begins from the address D000, and the lack part which begins

from the address E000 are buried with center subsoftware] case.

[0040] The group of the center subsoftware the object for the addresses A000 and for address C000 and the group of the center subsoftware the object for the addresses B000, the object for the addresses D000, and for address E000 here, It has completed as a function of subsoftware respectively, and if it is implementation which does not have influence in other portions of software, it will not matter even when it is functionally equivalent, even if it differs.

[0041] That the image of the application developed on a memory changes each time prepares some groups of equivalent center subsoftware functionally, and it is for reducing the danger of the analysis of the analysis from an image.

[0042] The center 1 side determines whether which group downloads, and the method is based on the random number calculation for example, by the side of the center 1. When there are generally N groups of center subsoftware, the number of 0 to N-1 is beforehand given to each class.
The random number by which it was generated in the random number generation part 13 is calculated by mod N, and the group of it compatible center subsoftware is downloaded.

[0043] The number is given to the group of each center subsoftware also when a function changes with groups of center subsoftware. Users' demand or the judgment by the side of the center 1 will determine the group of the subsoftware of which function the center 1 side transmits to the terminal 3 side.

[0044] In this case, when a fundamental function downloads by the first download and the demand from the terminal 3 is behind, the composition which downloads expanded function one by one is also possible.

[0045] Preparing some groups of functionally different center subsoftware, It is for downloading a function required of making only a different portion into center subsoftware, and measuring validation of resources simultaneously with protection of software about some applications also with a different portion, although it is alike.

[0046] Below, a number given to a group of each center subsoftware is called a center sub software number. The communication control part 31 of the terminal 3 recognizes of which center subsoftware a group downloaded, and this is used in order to determine whether to develop to a lack part of which position on a memory.

[0047] Below, a group of each center subsoftware describes a case where it has the same function mutually.

[0048] namely, download shown by a thick arrow in drawing 2 -- or if either of the downloads shown by an arrow of a dotted line is performed, application software which has the same function will be used.

[0049] A group of each center subsoftware, i.e., a group of center subsoftware an object for the addresses A000, and for address C000, It depends, for example on random number calculation by the side of the center 1 any of a group of center subsoftware an object for the addresses B000, an object for the addresses D000, and for address E000 download.

[0050] Beforehand, the former is made into No. 0, the latter is made into No. 1, and surplus calculation of the random number by which it was generated is done by mod 2. By 0 or 1, this chooses and downloads a group of a corresponding lack part of a number.

[0051] When the terminal 3 starts use of application software in the state of drawing 2, in order to judge whether they are that center subsoftware is stored in a lack part part, or empty area, there are the following methods.

[0052] For example, a code which carries out the return of the fixed values, such as "0", to specific fixed addresses, such as all the fields of a lack part part or leading addresses, is set up at the time of initial setting which stores terminal subsoftware in a memory of the terminal 3.

[0053] A code which carries out the return of the fixed values, such as "1", is set to a fixed address, for example, a leading address, and the last address of each block of center subsoftware.

[0054] When center subsoftware is transmitted to the terminal 3 from the center 1 and it is stored in a lack part by such setting out, the contents of the aforementioned lack part will be rewritten by center subsoftware. In terminal subsoftware within the terminal 3, data and a value by which the return was carried out from a fixed address in a storage area (the address A000 and address B000) of center subsoftware are investigated, If it is a fixed value (for example, "1") which shows that center subsoftware is stored, use of software will be continued, and software using will be stopped if it is a fixed value (for example, "0") which shows that it is a lack part.

[0055] For example, as a structure of terminal subsoftware, the address A000 and the address B000 can consider the following, when certainly returning either "0" or "1."
if A000= -- " -- 1"run from A000if A000= -- " -- 0"go to B000if B000= "1" -- run from B000if B=[000] "0" end -- by performing this setting out. Operation of application software can be suspended on the way, without a system performing unusual operation, when a part for operating normally is missing.

[0056] Furthermore, terminal subsoftware clears a portion which does not have necessity in operation after inner it of center subsoftware one by one working [software] .

[0057] For example, when center subsoftware downloads in drawing 2 to a lack part which begins from the address A000, and a lack part which begins from the address C000, Application software is used in the terminal 3, and if center subsoftware of a lack part which begins from the address A000 carries out the completion of use, center subsoftware of this lack part will be cleared. Thereby, if possible, the whole application software is made not to be developed on a memory simultaneously.

[0058] What, on the other hand, has the character which the maximum use count at the time of software using understands a priori as center subsoftware is set up, and a field which counts its own use count, respectively is provided. And if a count reaches the maximum use count set up a priori at the time of software using, it will be made structure which clears the center subsoftware. Thus, a portion unnecessary for use of software after inner [of center subsoftware] is clearable one by one from a memory.

[0059] There is the following as subsoftware with character which the maximum use count understands a priori.
(1) A thing of software for which more than a certain number of times is not used clearly constitutionally.

[0060] A portion etc. of an initializing function of software called only once [of the beginning] about one execution.
(2) A thing to make it use more than a certain number of times as a copyright management person's demand.

[0061] When collecting a charge for an animation, Still Picture Sub-Division, a sound, a game program, etc. by subsoftware called while using [one] software according to a use count, or when you would like to restrict a use count by trial software, etc. There are the following two in a clear method of the above center subsoftware.

[0062] One is the way the center subsoftware itself kills itself as shown in drawing 3. In this method, its clear command is described in front of a party of a last line of center subsoftware, and a command which returns to terminal subsoftware at a last line is written. By this method, only a last line will not be cleared but will remain.

[0063] Another is the way terminal subsoftware eliminates center subsoftware as shown in drawing 4.

[0064] When use of software finishes as mentioned above, all the downloaded center subsoftware is eliminated, and is initialized by state before download.

[0065] Being initialized here is returning a lack part to a state before center subsoftware's is developed. This means that a code which carries out the return of the fixed values, such as "0" etc. which shows that it is a lack part, to specific fixed addresses, such as all the fields of a lack part or leading addresses, is set up again.

[0066] Below, according to a sequence chart of drawing 5, an example of the above-mentioned whole system of operation is shown.

[0067] It is connected to the center 1 after starting, and terminal subsoftware sends a "center software requirement signal" to the center 1 (S101). After the center 1 which received a "center subsoftware requirement signal" generated a random number in the random number generation part 13, Based on a random number by which it was generated in the center subsoftware selecting part 12, it is determined of which center subsoftware a group is chosen (S102), From

the center subsoftware storage 11, it takes out with the memory development position, is made the terminal 3 with a "center subsoftware signal" through the communication control part 14, and downloads (S103).

[0068] The communication control part 31 of the terminal 3 develops and uses transmitted center subsoftware on a memory according to a center sub software number (S104).

[0069] When a group of another center subsoftware as expanded function is required, it downloads by sending a "center subsoftware requirement signal" to the center 1 again, and software is used (S105, S106, S107, S108).

[0070] Center subsoftware which became unnecessary while using software is cleared at any time during use. All things that were not eliminated after an end of software using by downloaded center subsoftware are cleared (S109). And a lack part with which center subsoftware was eliminated is initialized.

[0071]
[Effect of the Invention] Since according to this invention application software is divided and held at both a terminal and a center and the software on a terminal can operate independently, While the software is not operating, it is impossible to get to know the composition of the whole application software only by analyzing in a terminal.

[0072] When a terminal user operates application software, while certainly connecting with a center using a circuit, downloading the software on a center and combining with the software on a terminal, Software arrangement of the result set up and combined also differs each time so that the software to download may differ each time, and further, since the used portion on a memory is eliminated suitably, it makes software analysis difficult.

[0073] For this reason, without [without it complicates the structure of application software itself, and] enciphering, analysis of application software can be made difficult and the illegal use of the application software in a terminal can be prevented.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

TECHNICAL FIELD

[Industrial Application] This invention, So that software, such as an application program saved and executed at a terminal in the system which consists of a center and two or more terminals, may not be analyzed at a terminal. It is related with the software analysis protective method which protects and prevents the illegal use of software, such as an application program in a terminal.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

PRIOR ART

[Description of the Prior Art] Generally, to the software on a terminal, the illegal use (software analysis / alteration) by a malicious terminal user may occur. The illegal use of software begins from analyzing the software on a memory and a disk with tools, such as a debugger. In order to make this software analysis difficult, there is the method of complicating the software itself.

However, while the effect of this method is a problem of a grade, it cannot apply this method to short software, and cannot be said as perfectness. Although the mechanism in which the method of enciphering software decrypts the software is needed, since the mechanism itself cannot be enciphered, there is a fault that raw software will become clear, by analyzing the output portion.

[Translation done.]

* NOTICES *

JPO and INPIT are not responsible for any damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

EFFECT OF THE INVENTION

[Effect of the Invention] Since according to this invention application software is divided and held at both a terminal and a center and the software on a terminal can operate independently,

While the software is not operating, it is impossible to get to know the composition of the whole application software only by analyzing in a terminal.

[0072] When a terminal user operates application software, while certainly connecting with a center using a circuit, downloading the software on a center and combining with the software on a terminal, Software arrangement of the result set up and combined also differs each time so that the software to download may differ each time, and further, since the used portion on a memory is eliminated suitably, it makes software analysis difficult.

[0073] For this reason, without [without it complicates the structure of application software itself, and] enciphering, analysis of application software can be made difficult and the illegal use of the application software in a terminal can be prevented.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

TECHNICAL PROBLEM

[Problem to be solved by the invention] As mentioned above, in order to prevent the illegal use of the software on a terminal, can consider how to complicate the software itself and the method of enciphering, but. Since the mechanism in which there is a problem of being inapplicable in short software while the method of complicating is a problem of a grade, and the method of enciphering decodes software is needed and this mechanism itself cannot be enciphered, By analyzing the output portion, there is a problem that software will become clear and any method has a problem.

[0004] Without enciphering without complicating the software itself, the place which this invention was made in view of the above, and is made into the purpose makes the analysis difficult, and there is in providing the software analysis protective method which prevents the illegal use of software accurately.

[Translation done.]

* NOTICES *

JPO and INPIT are not responsible for any damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

MEANS

[Means for solving problem] In order to attain the above-mentioned purpose, this invention (Claim 1), In the system which comprised two or more terminals connected to the center and this center, It is a software analysis protective method protected so that the software used at a terminal may not be analyzed at a terminal, Distribute the software portion which divided the software used at a terminal to a terminal and a center, and it is held, Whenever it uses said software in a terminal, the software portion held from the terminal at the center is required, By loading the software portion which transmitted to the terminal the software portion held at the center according to the demand from a terminal, and was transmitted from the center to the missing part in the software portion held at the terminal on a memory, A software analysis protective method which consists of what operation of said software is enabled on a terminal, and said software developed on said memory is operated for on a terminal changing the arrangement pattern developed on said memory of said software whenever said software is used at a terminal.

[0006] In the above-mentioned software analysis protective method this invention (Claim 2), A center including the group of two or more software portions in which the software portion held at the center had a different function by choosing the group of one software portion which transmits to a terminal by judgment of the demand or center from a terminal, The function of said software to make it operate at a terminal is changed.

[0007] In the above-mentioned software analysis protective method this invention (Claim 3), A center including a group of two or more software portions with the function that a software portion held at a center is the same, by choosing a group of one software portion which transmits to a terminal based on a random number, An arrangement pattern developed on said memory of said software is changed whenever said software is used at a terminal.

[0008] In the above-mentioned software analysis protective method this invention (Claim 4), A software portion held at a center contains a group of two or more software portions, A center chooses a group of one software portion which transmits to a terminal, and when a group of this software portion transmitted from a center is loaded to a corresponding missing part on said memory, operation of said software is selectively attained at least on a terminal.

[0009] In the above-mentioned software analysis protective method, a center transmits this invention (Claim 5) one by one according to a demand from a terminal accompanying advance of operation of said software by a group of two or more of said software portions.

[0010] A software portion by which this invention (Claim 6) was loaded to a missing part in the above-mentioned software analysis protective method is cleared from said memory after an end

of use of said software.

[0011] In the above-mentioned software analysis protective method, this invention (Claim 7) is cleared from said memory one by one working [said software] as it becomes unnecessary [a software portion loaded to a missing part] for operation after said software.

[0012] This invention (Claim 8) is cleared from said memory in the above-mentioned software analysis protective method by command provided in a software portion in which a software portion loaded to a missing part was held at a terminal.

[0013] This invention (Claim 9) is cleared from said memory in the above-mentioned software analysis protective method by command provided in a software portion which a software portion loaded to a missing part was held at a center, and was loaded to a missing part.

[0014] In the above-mentioned software analysis protective method this invention (Claim 10), The maximum use count of each of this software portion of known [portion / each / which was held at a center / software] , It has a count of the number of times of actual use of each of this software portion, and this each software portion loaded to a missing part is cleared from said memory, when a count of said number of times of actual use reaches said maximum use count.

[0015] This invention (Claim 11) is characterized by software portions held at a terminal being two or more discontinuous blocks in said software in the above-mentioned software analysis protective method.

[0016] In the above-mentioned software analysis protective method this invention (Claim 12), a discontinuous block of a software portion which software portions held at a center are two or more discontinuous blocks in said software, and was held at a center -- this -- it is transmitted in different turn from an order between discontinuous block comrades

[0017] In the above-mentioned software analysis protective method this invention (Claim 13), It has a certain code which returns a certain fixed value which shows that each software portion held at a center is a software portion in which it was held at a center to a predetermined address in this each software portion, When each missing part on said memory has another code which returns another fixed value it indicates it to be that it is a missing part to a predetermined address in this each missing part and said another fixed value is returned working [said software] on a terminal, operation of said software is interrupted.

[0018] An analysis protective method this invention is characterized by that comprises the following.
This invention (Claim 14) is a center.
It is a software analysis protective method protected so that software used at a terminal may not be analyzed at a terminal in a system which comprised two or more terminals connected to this center, Distribute a software portion which divided software used at a terminal to a terminal and a center, and it is held, By loading a software portion which transmitted a software portion held at a center to a terminal, and was transmitted from a center to a missing part in a software portion held at a terminal on a memory, What operation of said software is enabled on a terminal, and

said software developed on said memory is operated for on a terminal

[0019] In the above-mentioned software analysis protective method, this invention (Claim 15) is transmitted to a terminal from a center according to the demand from a terminal, whenever the software portion held at the center uses said software in a terminal.

[0020] In the above-mentioned software analysis protective method, this invention (Claim 16) divides into two or more blocks the software portion held at the center, and is transmitted.

[0021] The arrangement pattern in which this invention (Claim 17) was developed on said memory of said software in the above-mentioned software analysis protective method is changed whenever said software is used at a terminal.

[0022] In the above-mentioned software analysis protective method this invention (Claim 18), The software portion held at the center contains the group of two or more software portions, A center chooses the group of one software portion which transmits to a terminal, and when the group of this software portion transmitted from the center is loaded to the corresponding missing part on said memory, operation of said software is selectively attained at least on a terminal.

[0023] The software portion by which this invention (Claim 19) was loaded to the missing part in the above-mentioned software analysis protective method is cleared from said memory after the end of use of said software.

[0024] In the above-mentioned software analysis protective method, this invention (Claim 20) is cleared from said memory one by one working [said software] as it becomes unnecessary [the software portion loaded to the missing part] for the operation after said software.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

OPERATION

[Function] In the software analysis protective method of this invention, it transmits to a terminal and the portion which divides and holds the software used at a terminal in both a terminal and the center, and holds the center at the time of software using is used. It is made to differ whenever it uses arrangement of the software used at the terminal on a memory at this time, Or the function of software is changed by a user's demand or judgment of a center, About what was lost in the

portion transmitted from the center among the software used at a terminal, the opportunity for a software image to be simultaneously developed on a memory by clearing one by one is made into the minimum. Furthermore, the portion to which after the end of use of software was always transmitted from the center is cleared.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

EXAMPLE

[Working example] Hereafter, the embodiment of this invention is described using Drawings.
[0027] Drawing 1 is a figure showing the composition of the system which enforces the software analysis protective method concerning one embodiment of this invention. This system comprises two or more terminals 3 connected with the center 1 and this center 1 via the public line 2.

[0028] Software, such as an application program used at each terminal 3, It is divided and saved in the terminal 3 and the center 1, and The inside of this software that is divided, is saved and is used at the terminal 3, A part of software which is saved to the terminal 3 and used at the terminal 3 will be called terminal subsoftware, and a part of software which is saved in the center 1 and used at the terminal 3 will be called center subsoftware. Here, center subsoftware is software which complements terminal subsoftware.

[0029] Here, it is aimed at all the software except the center 1 and the main part of the protocol control software which controls the data transfer between the terminals 3 for the software used at the terminal 3 mentioned above. For example, the data etc. which are used for the various application programs used at the terminal 3 or the program of the operation system level except the main part of transfer control, a database, etc. are meant, and these all serve as an applied object of this invention.

[0030] The system which consists of the center 1 to which this invention may be applied, and the terminal 3 may be a system which the case of a personal-computer-communications network also has, for example, and delivers application software from the center 1 side.

[0031] Especially this invention is suitable for a software circulation system which delivers to the terminal 3 for whenever [of use of an application software system / every] , and collects a usage fee.

[0032] As shown in drawing 1, The center 1 via the center subsoftware selecting part 12, the random number generation part 13, and the public line 2 which choose the center subsoftware stored in the center subsoftware storage 11 which stores center subsoftware, and this center subsoftware storage 11. Having [and] the communication control part 14 for communicating with the terminal 3, the terminal 3 has the communication control part 31 for communicating with the center 1 via the public line 2, and the terminal subsoftware memory 32.

[0033] In an initial state, terminal subsoftware vacates two or more fields of a certain amount of size as two or more discontinuous blocks, and is developed on the memory 32. Although drawing 2 shows this situation, an area (below, a lack part is called) which gave a slash which starts with the address A000, B000, C000, D000, and E000 hits it. Thus, since terminal subsoftware has a lack part, even if it performs only this terminal subsoftware, it does not carry out operation normal as application software.

[0034] Terminal subsoftware containing a lack part on the terminal 3 is distributed in a form of a floppy disk or CD-ROM, is loaded from a drive of the terminal 3, or has a method distributed by a file transfer from the center 1, for example. Or it memorizes beforehand on a hard disk of the terminal 3, and may be arranged on a memory at the time of system starting and software starting. Data of a size of an address of a lack part on a memory and an area of each part is beforehand set to the center 1 side here.

[0035] The software included in the lack part of terminal subsoftware is held as center subsoftware in the form of the discontinuous block at the center 1.
By loading these to the lack part of the terminal 3, application software will become perfect. However, the application software may operate, if it compatible center subsoftware does not go into all lack parts.

[0036] In the method of forming the lack part for all the center subsoftware, there is a method of transmitting all the center subsoftware collectively from the center 1 side, and there is also the method of making it the composition which operates even if it compatible center subsoftware does not go into all lack parts.

[0037] For example, if the software part of a basic function is stored on the memory of the terminal 3 at least when center subsoftware contains the software part of the basic function of application, and the software part of expanded function, There is a constitution method which becomes usable [the application software of the terminal 3] .

[0038] Or if a certain specific lack part (group) is buried, there is a constitution method which becomes usable [the application software of the terminal 3] .

[0039] Namely. [whether the lack part which begins from the address A000 like drawing 2, and the lack part which begins from the address C000 are buried with center subsoftware, and] Or it is the constitution method of operating in any [of whether the lack part which begins from the address B000, the lack part which begins from the address D000, and the lack part which begins from the address E000 are buried with center subsoftware] case.

[0040] The group of the center subsoftware the object for the addresses A000 and for address C000 and the group of the center subsoftware the object for the addresses B000, the object for the addresses D000, and for address E000 here, It has completed as a function of subsoftware respectively, and if it is implementation which does not have influence in other portions of software, it will not matter even when it is functionally equivalent, even if it differs.

[0041] That the image of the application developed on a memory changes each time prepares some groups of equivalent center subsoftware functionally, and it is for reducing the danger of the analysis of the analysis from an image.

[0042] The center 1 side determines whether which group downloads, and the method is based on the random number calculation for example, by the side of the center 1. When there are generally N groups of center subsoftware, the number of 0 to N-1 is beforehand given to each class.
The random number by which it was generated in the random number generation part 13 is calculated by mod N, and the group of it compatible center subsoftware is downloaded.

[0043] The number is given to the group of each center subsoftware also when a function changes with groups of center subsoftware. Users' demand or the judgment by the side of the center 1 will determine the group of the subsoftware of which function the center 1 side transmits to the terminal 3 side.

[0044] In this case, when a fundamental function downloads by the first download and the demand from the terminal 3 is behind, the composition which downloads expanded function one by one is also possible.

[0045] Preparing some groups of functionally different center subsoftware, It is for downloading a function required of making only a different portion into center subsoftware, and measuring validation of resources simultaneously with protection of software about some applications also with a different portion, although it is alike.

[0046] Below, the number given to the group of each center subsoftware is called a center sub software number. The communication control part 31 of the terminal 3 recognizes of which center subsoftware the group downloaded, and this is used in order to determine whether to develop to the lack part of which position on a memory.

[0047] Below, the group of each center subsoftware describes the case where it has the same function mutually.

[0048] namely, the download shown by a thick arrow in drawing 2 -- or if either of the downloads shown by the arrow of a dotted line is performed, application software which has the same function will be used.

[0049] The group of each center subsoftware, i.e., the group of the center subsoftware the object for the addresses A000, and for address C000, It depends, for example on the random number calculation by the side of the center 1 any of the group of the center subsoftware the object for

the addresses B000, the object for the addresses D000, and for address E000 download.

[0050] Beforehand, the former is made into No. 0, the latter is made into No. 1, and surplus calculation of the random number by which it was generated is done by mod 2. By 0 or 1, this chooses and downloads the group of the corresponding lack part of a number.

[0051] When the terminal 3 starts use of application software in the state of drawing 2, in order to judge whether they are that center subsoftware is stored in the lack part part, or empty area, there are the following methods.

[0052] For example, the code which carries out the return of the fixed values, such as "0", to specific fixed addresses, such as all the fields of a lack part part or leading addresses, is set up at the time of initial setting which stores terminal subsoftware in the memory of the terminal 3.

[0053] The code which carries out the return of the fixed values, such as "1", is set to the fixed address, for example, the leading address, and the last address of each block of center subsoftware.

[0054] When center subsoftware is transmitted to the terminal 3 from the center 1 and it is stored in a lack part by such setting out, the contents of the aforementioned lack part will be rewritten by center subsoftware. In the terminal subsoftware within the terminal 3, data and the value by which the return was carried out from a fixed address in the storage area (the address A000 and address B000) of center subsoftware are investigated, If it is a fixed value (for example, "1") which shows that center subsoftware is stored, use of software will be continued, and software using will be stopped if it is a fixed value (for example, "0") which shows that it is a lack part.

[0055] For example, as a structure of terminal subsoftware, the address A000 and the address B000 can consider the following, when certainly returning either "0" or "1."
if A000= -- " -- 1"run from A000if A000= -- " -- 0"go to B000if B000= "1" -- run from B000if B=[000] "0" end -- by performing this setting out. Operation of application software can be suspended on the way, without a system performing unusual operation, when the part for operating normally is missing.

[0056] Furthermore, terminal subsoftware clears the portion which does not have necessity in the operation after inner it of center subsoftware one by one working [software] .

[0057] For example, when center subsoftware downloads in drawing 2 to the lack part which begins from the address A000, and the lack part which begins from the address C000, Application software is used in the terminal 3, and if the center subsoftware of the lack part which begins from the address A000 carries out the completion of use, the center subsoftware of this lack part will be cleared. Thereby, if possible, the whole application software is made not to be developed on a memory simultaneously.

[0058] What, on the other hand, has the character which the maximum use count at the time of software using understands a priori as center subsoftware is set up, and the field which counts its own use count, respectively is provided. And if a count reaches the maximum use count set up a

priori at the time of software using, it will be made the structure which clears the center subsoftware. Thus, a portion unnecessary for use of the software after inner [of center subsoftware] is clearable one by one from a memory.

[0059] There is the following as subsoftware with character which the maximum use count understands a priori.
(1) The thing of software for which more than a certain number of times is not used clearly constitutionally.

[0060] The portion etc. of the initializing function of the software called only once [of the beginning] about one execution.
(2) A thing to make it use more than a certain number of times as a copyright management person's demand.

[0061] When collecting a charge for an animation, Still Picture Sub-Division, a sound, a game program, etc. by the subsoftware called while using [one] software according to a use count, or when you would like to restrict a use count by trial software, etc. There are the following two in the clear method of the above center subsoftware.

[0062] One is the way the center subsoftware itself kills itself as shown in drawing 3. In this method, its clear command is described in front of the party of the last line of center subsoftware, and the command which returns to terminal subsoftware at a last line is written. By this method, only a last line will not be cleared but will remain.
[0063] Another is the way terminal subsoftware eliminates center subsoftware as shown in drawing 4.
[0064] When use of software finishes as mentioned above, all the downloaded center subsoftware is eliminated, and is initialized by the state before download.

[0065] Being initialized here is returning a lack part to the state before center subsoftware's is developed. This means that the code which carries out the return of the fixed values, such as "0" etc. which shows that it is a lack part, to specific fixed addresses, such as all the fields of a lack part or leading addresses, is set up again.

[0066] Below, according to the sequence chart of drawing 5, the example of the above-mentioned whole system of operation is shown.

[0067] It is connected to the center 1 after starting, and terminal subsoftware sends a "center software requirement signal" to the center 1 (S101). After the center 1 which received the "center subsoftware requirement signal" generated the random number in the random number generation part 13, Based on the random number by which it was generated in the center subsoftware selecting part 12, it is determined of which center subsoftware a group is chosen (S102), From the center subsoftware storage 11, it takes out with the memory development position, is made the terminal 3 with a "center subsoftware signal" through the communication control part 14, and downloads (S103).

[0068] The communication control part 31 of the terminal 3 develops and uses the transmitted center subsoftware on a memory according to a center sub software number (S104).

[0069] When the group of another center subsoftware as expanded function is required, it downloads by sending a "center subsoftware requirement signal" to the center 1 again, and software is used (S105, S106, S107, S108).

[0070] The center subsoftware which became unnecessary while using software is cleared at any time during use. All the things that were not eliminated after the end of software using by the downloaded center subsoftware are cleared (S109). And the lack part with which center subsoftware was eliminated is initialized.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is a block diagram showing the composition of the system which enforces the software analysis protective method concerning one embodiment of this invention.

[Drawing 2] It is a mimetic diagram showing the relation between terminal subsoftware and center subsoftware in the above-mentioned embodiment.

[Drawing 3] In the above-mentioned embodiment, it is a mimetic diagram showing an example of the method of clearing center subsoftware.

[Drawing 4] In the above-mentioned embodiment, it is a mimetic diagram showing other examples of the method of clearing center subsoftware.

[Drawing 5] It is a sequence chart which shows the example of the whole system in the above-mentioned embodiment of operation.
[Explanations of letters or numerals]
1 Center
2 Public line
3 Terminal

11 Center subsoftware storage
12 Center subsoftware selecting part
13 Random number generation part
14 and 31 Communication control part
32 Terminal subsoftware memory

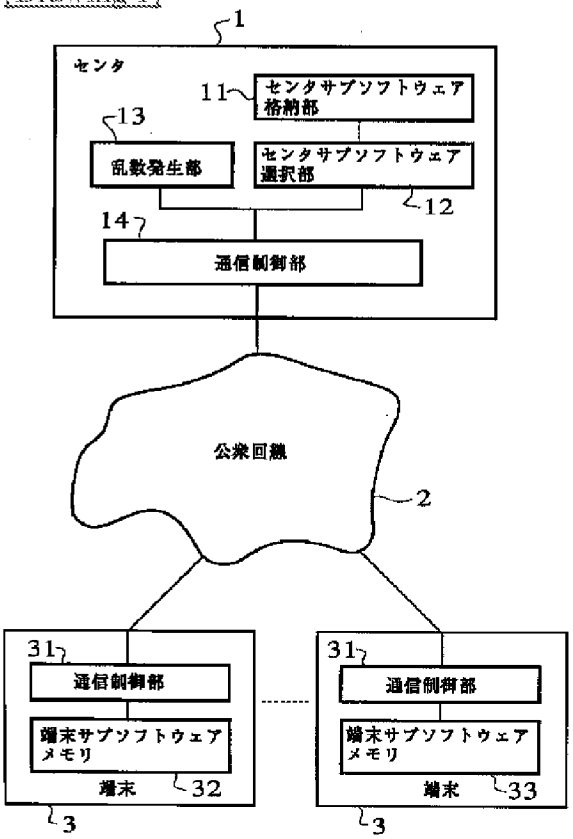
[Translation done.]

* NOTICES *

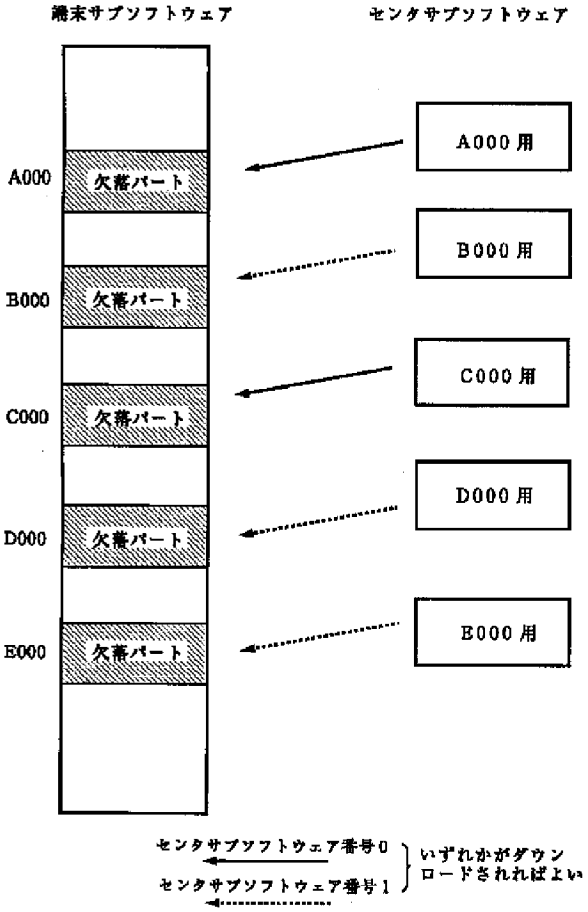
1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DRAWINGS

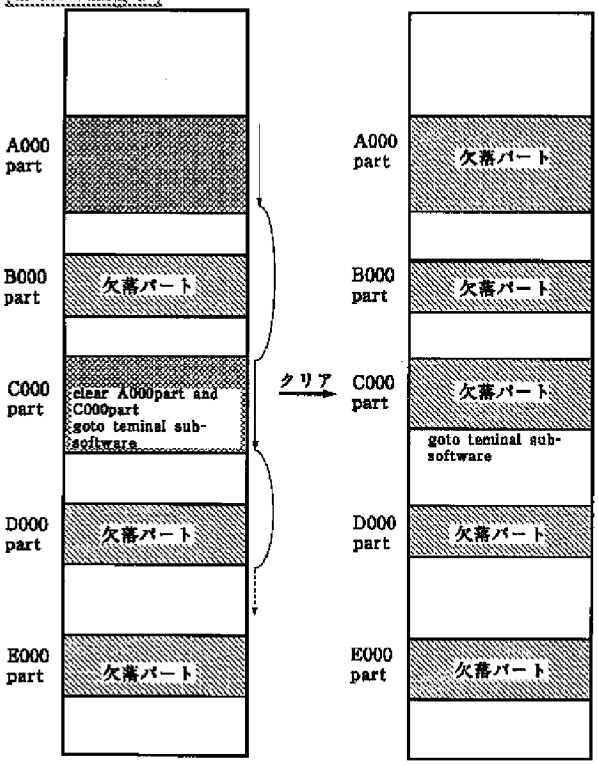
[Drawing 1]



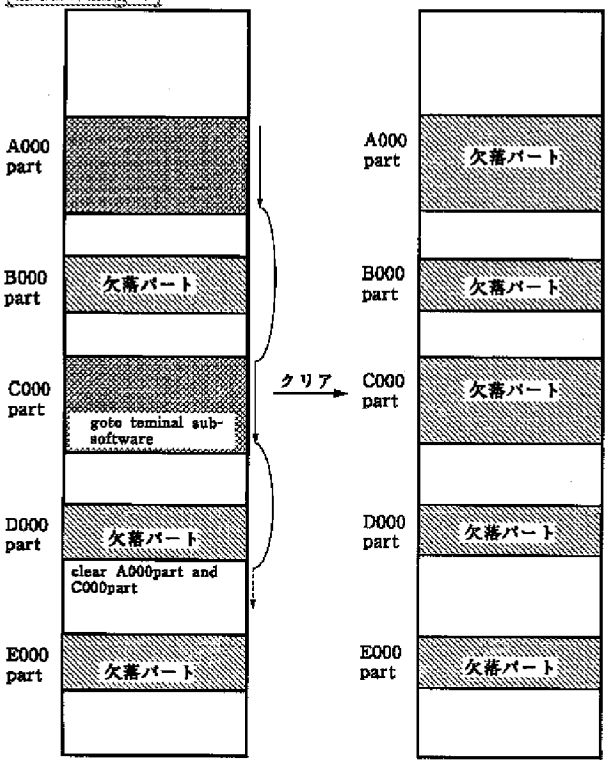
[Drawing 2]


端末サブソフトウェア
センタサブソフトウェア
A000
欠落パート
B000
欠落パート
C000
欠落パート
D000
欠落パート
E000
欠落パート
A000用
B000用
C000用
D000用
E000用
センタサブソフトウェア番号0
センタサブソフトウェア番号1
いずれかがダウンロードされればよい

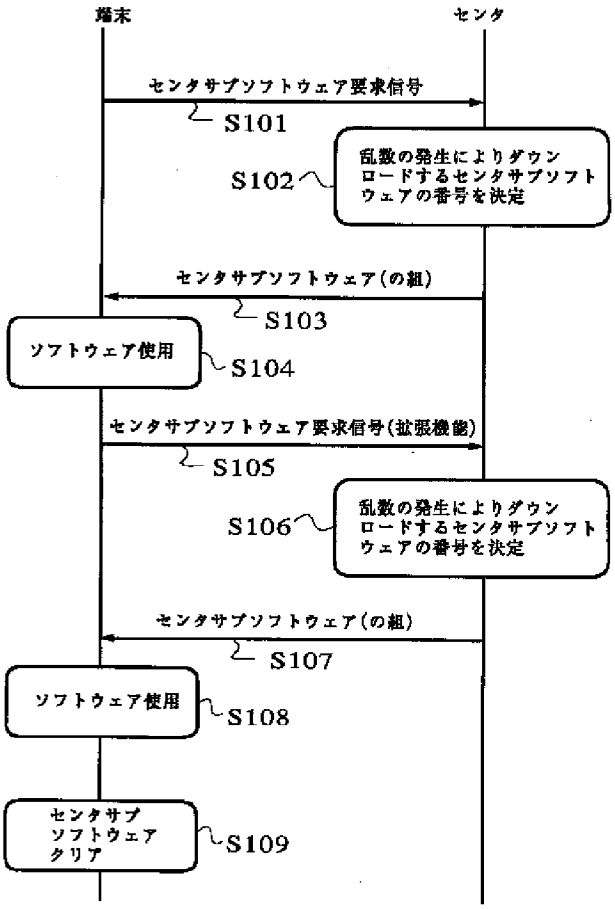
[Drawing 3]


A000 part
B000 part
C000 part
clear A000part and C000part goto teminal sub-software
D000 part
E000 part
欠落パート
クリア
A000 part
B000 part
C000 part
D000 part
E000 part
欠落パート
goto teminal sub-software

[Drawing 4]



[Drawing 5]



[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平8－16385

(43)公開日　平成８年(1996)１月19日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/06	５５０　Ｘ	7230－5B		
15/00	３３０　Ａ	9364－5L		

審査請求　未請求　請求項の数20　ＯＬ　（全 10 頁）

(21)出願番号　特願平7－105177

(22)出願日　平成７年(1995)４月28日

(31)優先権主張番号　特願平6－92134

(32)優先日　平６(1994)４月28日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号

(72)発明者　奥山　浩伸
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内

(72)発明者　森保　健治
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内

(72)発明者　金井　敦
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内

(74)代理人　弁理士　三好　秀和　（外１名）

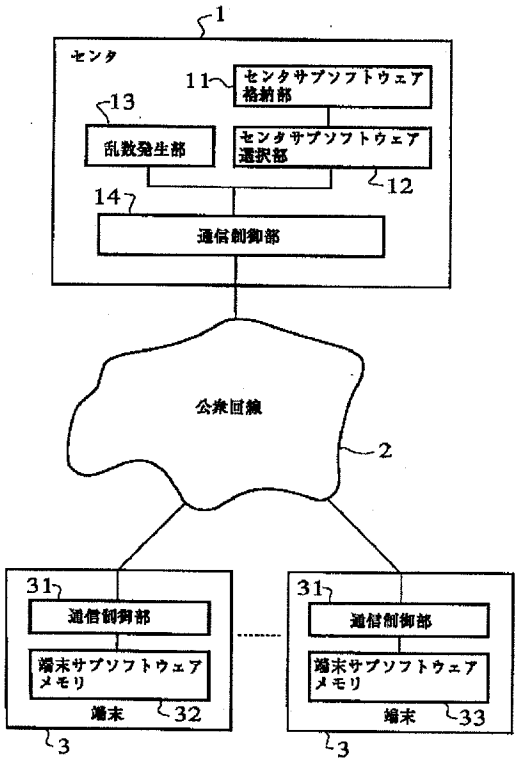
最終頁に続く

(54)【発明の名称】　ソフトウェア解析保護方法

(57)【要約】

【目的】　ソフトウェア自体を複雑にすることなく、また暗号化することもなく、その解析を困難にし、ソフトウェアの不正利用を適確に防止するソフトウェア解析保護方法を提供する。

【構成】　アプリケーションソフトウェアをセンタ１と端末３の両方で分割して、それぞれセンタサブソフトウェア格納部１１および端末サブソフトウェアメモリ３２に保持し、端末３にダウンロードするアプリケーションソフトウェアの欠落パートをセンタ１の乱数発生部１３およびセンタサブソフトウェア選択部１２によって適宜選択して、端末３にダウンロードすることにより、端末３におけるメモリ上のアプリケーションソフトウェアの配置を使用する度に異なるように設定し、すべてのアプリケーションソフトウェアが同時にメモリ上に展開されないようにする。



1

【特許請求の範囲】

【請求項１】　センタと、該センタに接続された複数の端末で構成されたシステムにおいて、端末で使用されるソフトウェアが端末で解析されないように保護するソフトウェア解析保護方法であって、

端末で使用されるソフトウェアを分割したソフトウェア部分を端末とセンタとに分配して保持し、

端末において前記ソフトウェアを使用する毎に、端末からセンタに保持されたソフトウェア部分を要求し、

端末からの要求に応じてセンタに保持されたソフトウェア部分を端末に送信し、センタから送信されたソフトウェア部分をメモリ上の端末に保持されたソフトウェア部分中の欠落部分にロードすることにより、前記ソフトウェアの前記メモリ上に展開された配置パターンを前記ソフトウェアが端末で使用される毎に変更しながら、前記ソフトウェアを端末上で動作可能とし、

前記メモリ上に展開された前記ソフトウェアを端末上で動作させる、

ことからなるソフトウェア解析保護方法。

【請求項２】　請求項１記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は異なる機能を持った複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を端末からの要求又はセンタの判断により選択することにより、端末で動作させる前記ソフトウェアの機能を変更することを特徴とするもの。

【請求項３】　請求項１記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は同じ機能を持った複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を乱数に基づいて選択することにより、前記ソフトウェアの前記メモリ上に展開された配置パターンを前記ソフトウェアが端末で使用される毎に変更することを特徴とするもの。

【請求項４】　請求項１記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を選択し、前記ソフトウェアはセンタから送信された該ソフトウェア部分の組が前記メモリ上の対応する欠落部分にロードされた時に端末上で少なくとも部分的に動作可能となることを特徴とするもの。

【請求項５】　請求項１記載のソフトウェア解析保護方法において、センタは前記複数のソフトウェア部分の組を、前記ソフトウェアの動作の進行に伴う端末からの要求に応じて順次送信することを特徴とするもの。

【請求項６】　請求項１記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの使用終了後に前記メモリからクリアされることを特徴とするもの。

2

【請求項７】　請求項１記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの以降の動作に不必要となるに従い前記ソフトウェアの動作中に順次前記メモリからクリアされることを特徴とするもの。

【請求項８】　請求項１記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は端末に保持されたソフトウェア部分に設けられたコマンドにより前記メモリからクリアされることを特徴とするもの。

【請求項９】　請求項１記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分はセンタに保持され欠落部分にロードされたソフトウェア部分に設けられたコマンドにより前記メモリからクリアされることを特徴とするもの。

【請求項１０】　請求項１記載のソフトウェア解析保護方法において、センタに保持された各ソフトウェア部分は既知の該各ソフトウェア部分の最大使用回数と、該各ソフトウェア部分の実使用回数のカウントとを有し、欠落部分にロードされた該各ソフトウェア部分は前記実使用回数のカウントが前記最大使用回数に達した時に前記メモリからクリアされることを特徴とするもの。

【請求項１１】　請求項１記載のソフトウェア解析保護方法において、端末に保持されたソフトウェア部分は前記ソフトウェア中の複数の非連続なブロックであることを特徴とするもの。

【請求項１２】　請求項１記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は前記ソフトウェア中の複数の非連続なブロックであり、センタに保持されたソフトウェア部分の非連続なブロックは該非連続なブロック同志間の順序と異なる順番で送信されることを特徴とするもの。

【請求項１３】　請求項１記載のソフトウェア解析保護方法において、センタに保持された各ソフトウェア部分はそれがセンタに保持されたソフトウェア部分であることを示す或る固定値を返す或るコードを該各ソフトウェア部分内の所定のアドレスに有し、前記メモリ上の各欠落部分はそれが欠落部分であることを示す別の固定値を返す別のコードを該各欠落部分内の所定のアドレスに有し、端末上で前記ソフトウェアの動作中に前記別の固定値が返された時に前記ソフトウェアの動作が中断されることを特徴とするもの。

【請求項１４】　センタと、該センタに接続された複数の端末で構成されたシステムにおいて、端末で使用されるソフトウェアが端末で解析されないように保護するソフトウェア解析保護方法であって、

端末で使用されるソフトウェアを分割したソフトウェア部分を端末とセンタとに分配して保持し、

センタに保持されたソフトウェア部分を端末に送信し、センタから送信されたソフトウェア部分をメモリ上の端

末に保持されたソフトウェア部分中の欠落部分にロードすることにより、前記ソフトウェアを端末上で動作可能とし、

前記メモリ上に展開された前記ソフトウェアを端末上で動作させる、ことからなるソフトウェア解析保護方法。

【請求項１５】　請求項１４記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は端末において前記ソフトウェアを使用する毎に、端末からの要求に応じてセンタから端末に送信されることを特徴とするもの。

【請求項１６】　請求項１４記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のブロックに分けて送信されることを特徴とするもの。

【請求項１７】　請求項１４記載のソフトウェア解析保護方法において、前記ソフトウェアの前記メモリ上に展開された配置パターンは前記ソフトウェアが端末で使用される毎に変更されることを特徴とするもの。

【請求項１８】　請求項１４記載のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を選択し、前記ソフトウェアはセンタから送信された該ソフトウェア部分の組が前記メモリ上の対応する欠落部分にロードされた時に端末上で少なくとも部分的に動作可能となることを特徴とするもの。

【請求項１９】　請求項１４記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの使用終了後に前記メモリからクリアされることを特徴とするもの。

【請求項２０】　請求項１４記載のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの以降の動作に不必要となるに従い前記ソフトウェアの動作中に順次前記メモリからクリアされることを特徴とするもの。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、センタと複数の端末からなるシステムにおいて端末で保存され実行されるアプリケーションプログラム等のソフトウェアが端末で解析されないように保護して端末におけるアプリケーションプログラム等のソフトウェアの不正利用を防止するソフトウェア解析保護方法に関する。

【０００２】

【従来の技術】一般に、端末上にあるソフトウェアに対しては、悪意のある端末使用者による不正利用（ソフトウェア解析／改ざん）がありうる。ソフトウェアの不正利用は、デバッガなどのツールによってメモリ上、ディスク上のソフトウェアを解析することから始まる。このソフトウェア解析を困難にするには、ソフトウェア自体

を複雑にする方法がある。しかしながら、この方法の効力は程度の問題であると同時に、この方法は短いソフトウェアに適用することができず、万全とはいえない。また、ソフトウェアを暗号化する方法は、そのソフトウェアを復号化する機構が必要となるが、その機構自身は暗号化することができないため、その出力部分を解析することにより、生のソフトウェアが明らかになってしまうという欠点がある。

【０００３】

【発明が解決しようとする課題】上述したように、端末上のソフトウェアの不正利用を防止するために、ソフトウェア自体を複雑にする方法や暗号化する方法が考えられるが、複雑化する方法は程度の問題であるとともに短いソフトウェアには適用することができないという問題があり、また暗号化する方法はソフトウェアを複合化する機構が必要となり、この機構自身は暗号化できないため、その出力部分を解析することによりソフトウェアが明らかになってしまうという問題があり、いずれの方法も問題がある。

【０００４】本発明は、上記に鑑みてなされたもので、その目的とするところは、ソフトウェア自体を複雑にすることなく、また暗号化することもなく、その解析を困難にし、ソフトウェアの不正利用を適確に防止するソフトウェア解析保護方法を提供することにある。

【０００５】

【課題を解決するための手段】上記目的を達成するため、本発明（請求項１）は、センタと、該センタに接続された複数の端末で構成されたシステムにおいて、端末で使用されるソフトウェアが端末で解析されないように保護するソフトウェア解析保護方法であって、端末で使用されるソフトウェアを分割したソフトウェア部分を端末とセンタとに分配して保持し、端末において前記ソフトウェアを使用する毎に、端末からセンタに保持されたソフトウェア部分を要求し、端末からの要求に応じてセンタに保持されたソフトウェア部分を端末に送信し、センタから送信されたソフトウェア部分をメモリ上の端末に保持されたソフトウェア部分中の欠落部分にロードすることにより、前記ソフトウェアの前記メモリ上に展開された配置パターンを前記ソフトウェアが端末で使用される毎に変更しながら、前記ソフトウェアを端末上で動作可能とし、前記メモリ上に展開された前記ソフトウェアを端末上で動作させる、ことからなるソフトウェア解析保護方法。

【０００６】又、本発明（請求項２）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は異なる機能を持った複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を端末からの要求又はセンタの判断により選択することにより、端末で動作させる前記ソフトウェアの機能を変更することを特徴とする。

5

【０００７】又、本発明（請求項３）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は同じ機能を持った複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を乱数に基づいて選択することにより、前記ソフトウェアの前記メモリ上に展開された配置パターンを前記ソフトウェアが端末で使用される毎に変更することを特徴とする。

【０００８】又、本発明（請求項４）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を選択し、前記ソフトウェアはセンタから送信された該ソフトウェア部分の組が前記メモリ上の対応する欠落部分にロードされた時に端末上で少なくとも部分的に動作可能となることを特徴とする。

【０００９】又、本発明（請求項５）は、上記のソフトウェア解析保護方法において、センタは前記複数のソフトウェア部分の組を、前記ソフトウェアの動作の進行に伴う端末からの要求に応じて順次送信することを特徴とする。

【００１０】又、本発明（請求項６）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの使用終了後に前記メモリからクリアされることを特徴とする。

【００１１】又、本発明（請求項７）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの以降の動作に不必要となるに従い前記ソフトウェアの動作中に順次前記メモリからクリアされることを特徴とする。

【００１２】又、本発明（請求項８）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は端末に保持されたソフトウェア部分に設けられたコマンドにより前記メモリからクリアされることを特徴とする。

【００１３】又、本発明（請求項９）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分はセンタに保持され欠落部分にロードされたソフトウェア部分に設けられたコマンドにより前記メモリからクリアされることを特徴とする。

【００１４】又、本発明（請求項１０）は、上記のソフトウェア解析保護方法において、センタに保持された各ソフトウェア部分は既知の該各ソフトウェア部分の最大使用回数と、該各ソフトウェア部分の実使用回数のカウントとを有し、欠落部分にロードされた該各ソフトウェア部分は前記実使用回数のカウントが前記最大使用回数に達した時に前記メモリからクリアされることを特徴とする。

【００１５】又、本発明（請求項１１）は、上記のソフトウェア解析保護方法において、端末に保持されたソフトウェア部分は前記ソフトウェア中の複数の非連続なブロックであることを特徴とする。

6

【００１６】又、本発明（請求項１２）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は前記ソフトウェア中の複数の非連続なブロックであり、センタに保持されたソフトウェア部分の非連続なブロックは該非連続なブロック同志間の順序と異なる順番で送信されることを特徴とする。

【００１７】又、本発明（請求項１３）は、上記のソフトウェア解析保護方法において、センタに保持された各ソフトウェア部分はそれがセンタに保持されたソフトウェア部分であることを示す或る固定値を返す或るコードを該各ソフトウェア部分内の所定のアドレスに有し、前記メモリ上の各欠落部分はそれが欠落部分であることを示す別の固定値を返す別のコードを該各欠落部分内の所定のアドレスに有し、端末上で前記ソフトウェアの動作中に前記別の固定値が返された時に前記ソフトウェアの動作が中断されることを特徴とする。

【００１８】更に本発明は、本発明（請求項１４）は、センタと、該センタに接続された複数の端末で構成されたシステムにおいて、端末で使用されるソフトウェアが端末で解析されないように保護するソフトウェア解析保護方法であって、端末で使用されるソフトウェアを分割したソフトウェア部分を端末とセンタとに分配して保持し、センタに保持されたソフトウェア部分を端末に送信し、センタから送信されたソフトウェア部分をメモリ上の端末に保持されたソフトウェア部分中の欠落部分にロードすることにより、前記ソフトウェアを端末上で動作可能とし、前記メモリ上に展開された前記ソフトウェアを端末上で動作させる、ことからなるソフトウェア解析保護方法を提供する。

【００１９】又、本発明（請求項１５）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は端末において前記ソフトウェアを使用する毎に、端末からの要求に応じてセンタから端末に送信されることを特徴とする。

【００２０】又、本発明（請求項１６）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のブロックに分けて送信されることを特徴とする。

【００２１】又、本発明（請求項１７）は、上記のソフトウェア解析保護方法において、前記ソフトウェアの前記メモリ上に展開された配置パターンは前記ソフトウェアが端末で使用される毎に変更されることを特徴とする。

【００２２】又、本発明（請求項１８）は、上記のソフトウェア解析保護方法において、センタに保持されたソフトウェア部分は複数のソフトウェア部分の組を含み、センタは端末に送信する一つのソフトウェア部分の組を選択し、前記ソフトウェアはセンタから送信された該ソ

フトウェア部分の組が前記メモリ上の対応する欠落部分にロードされた時に端末上で少なくとも部分的に動作可能となることを特徴とする。

【００２３】又、本発明（請求項１９）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの使用終了後に前記メモリからクリアされることを特徴とする。

【００２４】又、本発明（請求項２０）は、上記のソフトウェア解析保護方法において、欠落部分にロードされたソフトウェア部分は前記ソフトウェアの以降の動作に不必要となるに従い前記ソフトウェアの動作中に順次前記メモリからクリアされることを特徴とする。

【００２５】

【作用】本発明のソフトウェア解析保護方法では、端末で使用されるソフトウェアを端末とセンタの両方で分割して保持し、ソフトウェア使用時にセンタの保持している部分を端末に送信し使用する。このときメモリ上の端末で使用されるソフトウェアの配置を使用する度に異ならせ、あるいはユーザの要求やセンタの判断によりソフトウェアの機能を変更し、また端末で使用されるソフトウェアのうちセンタから送信された部分で必要のなくなったものについては逐次クリアすることで同時にメモリ上にソフトウェアイメージが展開される機会を最小限にする。さらにソフトウェアの使用終了後は必ずセンタから送信された部分はクリアする。

【００２６】

【実施例】以下、図面を用いて本発明の実施例を説明する。

【００２７】図１は、本発明の一実施例に係わるソフトウェア解析保護方法を実施するシステムの構成を示す図である。本システムは、センタ１と、該センタ１と公衆回線２を介して接続された複数の端末３とから構成されている。

【００２８】各端末３で使用されるアプリケーションプログラム等のソフトウェアは、端末３とセンタ１とに分割されて保存され、この分割されて保存され端末３で使用されるソフトウェアのうち、端末３に保存され端末３で使用されるソフトウェアの一部を端末サブソフトウェアと称し、センタ１に保存され端末３で使用されるソフトウェアの一部をセンタサブソフトウェアと称することにする。ここで、センタサブソフトウェアは、端末サブソフトウェアを補完するソフトウェアである。

【００２９】ここで、上述した端末３で使用されるソフトウェアとは、センタ１と端末３間のデータ転送を制御するプロトコル制御ソフトウェアの主要部分を除く全てのソフトウェアを対象とする。例えば、端末３で使用される各種アプリケーションプログラム、あるいは転送制御の主要部分を除くオペレーションシステムレベルのプログラム、データベースなどに使用されるデータ等を意味しており、これらは全て本発明の適用対象となる。

【００３０】また本発明が適用され得るセンタ１と端末３とからなるシステムとは、例えばパソコン通信ネットワークの場合もあり、センタ１側からアプリケーションソフトウェアを配送するシステムである場合もある。

【００３１】特に本発明は、アプリケーションソフトウェアシステムを使用のたび毎に端末３に配送し、使用料を徴収するようなソフトウェア流通システムに好適なものである。

【００３２】図１に示すように、センタ１はセンタサブソフトウェアを格納するセンタサブソフトウェア格納部１１、このセンタサブソフトウェア格納部１１に格納されたセンタサブソフトウェアを選択するセンタサブソフトウェア選択部１２、乱数発生部１３、および公衆回線２を介して端末３と通信するための通信制御部１４を有し、また端末３は公衆回線２を介してセンタ１と通信するための通信制御部３１および端末サブソフトウェアメモリ３２を有する。

【００３３】初期状態において、端末サブソフトウェアは、複数の非連続なブロックとしてある程度の大きさの領域を複数箇所空けてメモリ３２上に展開される。図２はこの様子を示しているが、番地Ａ０００，Ｂ０００，Ｃ０００，Ｄ０００，Ｅ０００で始まる斜線を施したエリア（以下では、欠落パートと称する）がそれに当たる。このように、端末サブソフトウェアは欠落パートを有しているため、この端末サブソフトウェアだけを実行しても、アプリケーションソフトウェアとしては正常な動作をしない。

【００３４】端末３上の欠落パートを含む端末サブソフトウェアは、例えば、フロッピーディスクやＣＤ－ＲＯＭの形で配布され端末３のドライブからロードしたり、センタ１からのファイル転送で配布される方法がある。あるいは端末３のハードディスク上にあらかじめ記憶しておき、システム立上げ時やソフトウェア起動時にメモリ上に配置される場合もある。なお、ここでメモリ上の欠落パートのアドレスと各パートのエリアの大きさのデータは、センタ１側にあらかじめ設定されている。

【００３５】端末サブソフトウェアの欠落パートに入るソフトウェアは、センタ１に非連続なブロックの形でセンタサブソフトウェアとして保持されており、これらが端末３の欠落パートにロードされることにより、アプリケーションソフトウェアは完全なものとなる。しかしアプリケーションソフトウェアは、全ての欠落パートにそれに対応するセンタサブソフトウェアが入らなければ動作しないという訳ではない。

【００３６】全センタサブソフトウェア分の欠落パートを設ける方法では、センタ１側から全センタサブソフトウェアを一括して転送する方法もあるが、全ての欠落パートにそれに対応するセンタサブソフトウェアが入らなくても動作する構成にする方法もある。

【００３７】例えば、センタサブソフトウェアが、アプ

リケーションの基本機能のソフトウェア部と拡張機能のソフトウェア部とを含んでいた場合、少なくとも基本機能のソフトウェア部が端末３のメモリ上に格納されれば、端末３のアプリケーションソフトウェアが使用可能となる構成方法がある。

【００３８】あるいは、ある特定の欠落パート（の組）が埋められれば、端末３のアプリケーションソフトウェアは使用可能となる構成方法がある。

【００３９】すなわち、図２のように番地Ａ０００から始まる欠落パートと番地Ｃ０００から始まる欠落パートとがセンタサブソフトウェアで埋まるか、あるいは番地Ｂ０００から始まる欠落パートと番地Ｄ０００から始まる欠落パートと番地Ｅ０００から始まる欠落パートとがセンタサブソフトウェアで埋まるかのいずれの場合にも動作する、という構成方法である。

【００４０】ここで番地Ａ０００用と番地Ｃ０００用のセンタサブソフトウェアの組と、番地Ｂ０００用と番地Ｄ０００用と番地Ｅ０００用のセンタサブソフトウェアの組とは、各々サブソフトウェアの機能として完結しており、ソフトウェアの他の部分に影響がないインプリメントであれば、機能的には同等でも異なっていても構わない。

【００４１】機能的に同等なセンタサブソフトウェアの組をいくつか用意するのは、メモリ上に展開されるアプリケーションのイメージが毎回変わることで、イメージからの解析の解析の危険性を低減するためである。

【００４２】いずれの組がダウンロードされるかは、センタ１側が決定し、その方法は例えばセンタ１側の乱数計算による。一般にセンタサブソフトウェアの組がＮ個ある場合は、あらかじめ各組に０からＮ－１の番号が与えられており、乱数発生部１３で発生した乱数をｍｏｄ　Ｎで計算し、それに対応するセンタサブソフトウェアの組をダウンロードする。

【００４３】センタサブソフトウェアの組によって機能が異なる場合も各センタサブソフトウェアの組には番号が与えられている。センタ１側がどの機能のサブソフトウェアの組を端末３側に送信するかをユーザ側の要求、あるいはセンタ１側の判断により決定することになる。

【００４４】この場合、最初のダウンロードで基本的な機能がダウンロードされ、後に端末３からの要求があった場合、順次拡張機能をダウンロードしていく構成も可能である。

【００４５】機能的に異なるセンタサブソフトウェアの組をいくつか用意するのは、似ているが異なった部分も持ついくつかのアプリケーションについて、異なった部分のみをセンタサブソフトウェアとすることで必要な機能をダウンロードして、ソフトウェアの保護と同時に資源の有効化を計るためである。

【００４６】各センタサブソフトウェアの組に与えられた番号を以下ではセンタサブソフトウェア番号と呼ぶ。これは端末３の通信制御部３１が、どのセンタサブソフトウェアの組がダウンロードされたかを認識し、メモリ上のどの位置の欠落パートに展開するかを決定するために使用される。

【００４７】以下では各センタサブソフトウェアの組が、互いに同じ機能を持つ場合について述べる。

【００４８】すなわち図２において、太い矢印で示すダウンロードかまたは点線の矢印で示すダウンロードのいずれかが行なわれれば同じ機能を持つようなアプリケーションソフトウェアが使用される。

【００４９】それぞれのセンタサブソフトウェアの組、すなわち番地Ａ０００用と番地Ｃ０００用のセンタサブソフトウェアの組と、番地Ｂ０００用と番地Ｄ０００用と番地Ｅ０００用のセンタサブソフトウェアの組の、いずれがダウンロードされるかは、例えばセンタ１側の乱数計算による。

【００５０】あらかじめ前者を０番、後者を１番としておき、発生した乱数をｍｏｄ　２で剰余計算する。これが０か１かにより、対応する番号の欠落パートの組を選びダウンロードする。

【００５１】図２の状態で、端末３がアプリケーションソフトウェアの使用を開始した場合、欠落パート部にセンタサブソフトウェアが格納されているのか、あるいは空きエリアであるのかを判定するためには以下の方法がある。

【００５２】例えば、端末サブソフトウェアを端末３のメモリに格納する初期設定時、欠落パート部の全ての領域あるいは先頭番地などの特定の固定番地に“０”などの固定値をリターンするコードを設定しておく。

【００５３】又、センタサブソフトウェアの各ブロックの固定番地、例えば先頭番地や最終番地などには、“１”などの固定値をリターンするコードを設定しておく。

【００５４】このような設定により、センタ１から端末３へセンタサブソフトウェアが転送され欠落パートに格納されると、前記の欠落パートの内容がセンタサブソフトウェアに書き換えられることとなる。端末３内の端末サブソフトウェアでは、センタサブソフトウェアの格納エリア（番地Ａ０００や番地Ｂ０００）内の固定番地からのデータやリターンされた値を調べ、センタサブソフトウェアが格納されていることを示す固定値（例えば“１”）であればソフトウェアの使用を続行し、欠落パートであることを示す固定値（例えば“０”）であればソフトウェア使用を中止する。

【００５５】例えば、端末サブソフトウェアの構造としては、番地Ａ０００，番地Ｂ０００は必ず”０”，”１”のいずれかを返すとした場合、以下のものが考えられる。

ｉｆ　Ａ０００＝“１”　ｒｕｎ　ｆｒｏｍ　Ａ０００
ｉｆ　Ａ０００＝“０”　ｇｏ　ｔｏ　Ｂ０００

ｉｆ　Ｂ０００＝“１”　ｒｕｎ　ｆｒｏｍ　Ｂ０００
ｉｆ　Ｂ０００＝“０”　ｅｎｄ

この設定を行なうことで、正常に動作するためのパートが欠落している場合に、システムが異常な動作を行なうこと無しに、アプリケーションソフトウェアの動作を途中で停止することができる。

【００５６】さらに端末サブソフトウェアは、センタサブソフトウェアの内それ以降の動作に必要のない部分をソフトウェアの動作中に逐次クリアする。

【００５７】例えば図２において、センタサブソフトウェアが番地Ａ０００から始まる欠落パートと番地Ｃ０００から始まる欠落パートとにダウンロードされたとき、端末３においてアプリケーションソフトウェアが使用され、番地Ａ０００から始まる欠落パートのセンタサブソフトウェアが使用完了すると、この欠落パートのセンタサブソフトウェアはクリアされる。これによりアプリケーションソフトウェア全体がなるべくメモリ上に同時に展開されないようにする。

【００５８】一方、センタサブソフトウェアとして、ソフトウェア使用時の最大使用回数が事前に分かるような性質を持つものを設定しておくと共に、それぞれ自分自身の使用回数をカウントする領域を設ける。そしてソフトウェア使用時に、カウントが事前に設定された最大使用回数に達すると、そのセンタサブソフトウェアをクリアする構造にしておく。このようにして、センタサブソフトウェアの内以降のソフトウェアの使用に不必要な部分をメモリ上から順次クリアすることができる。

【００５９】最大使用回数が事前に分かるような性質を持つサブソフトウェアとしては次のようなものがある。

（１）ソフトウェアの構成上、明らかにある回数以上は使用されないもの。

【００６０】一回の実行について冒頭の一回だけ呼び出されるソフトウェアの初期化機能の部分など。

（２）著作権管理者の要求として、ある回数以上は使用させたくないもの。

【００６１】一つのソフトウェアの使用中に呼び出されるサブソフトウェアで、動画・静止画・音声・ゲームプログラム等を使用回数に応じて料金を徴収する場合、あるいは試用ソフトウェアで使用回数を制限したい場合など。尚、上記のようなセンタサブソフトウェアのクリアの方法には次の二つがある。

【００６２】一つは図３に示すようにセンタサブソフトウェア自身が自分を消す方法である。この方法では、センタサブソフトウェアの最終行の一行前に自分自身のクリア命令が記述されており、最終行に端末サブソフトウェアへ戻る命令が書かれている。この方法では最終行のみはクリアされず残ることとなる。

【００６３】もう一つは図４に示すように端末サブソフトウェアがセンタサブソフトウェアを消去する方法である。

【００６４】以上のようにしてソフトウェアの使用が終わった時、ダウンロードされた全てのセンタサブソフトウェアは消去され、ダウンロード前の状態に初期化される。

【００６５】ここで初期化されるとは、欠落パートをセンタサブソフトウェアが展開される前の状態に戻すことである。これは例えば、欠落パートの全ての領域あるいは先頭番地等の特定の固定番地に、欠落パートであることを示す“０”などの固定値をリターンするコードが再び設定されることを意味する。

【００６６】以下では、図５のシーケンスチャートに従い上記のシステム全体の動作例を示す。

【００６７】端末サブソフトウェアは起動後、センタ１に接続され、”センタソフトウェア要求信号”をセンタ１に送る（Ｓ１０１）。”センタサブソフトウェア要求信号”を受信したセンタ１は乱数発生部１３で乱数を発生させた後、センタサブソフトウェア選択部１２にて発生した乱数に基づき、どのセンタサブソフトウェアの組を選択するかを決定し（Ｓ１０２）、センタサブソフトウェア格納部１１から、そのメモリ展開位置とともに取り出し、通信制御部１４を通して端末３に“センタサブソフトウェア信号”としダウンロードする（Ｓ１０３）。

【００６８】端末３の通信制御部３１は、送信されてきたセンタサブソフトウェアを、センタサブソフトウェア番号に応じてメモリ上に展開し、使用する（Ｓ１０４）。

【００６９】拡張機能としての別のセンタサブソフトウェアの組が必要な時は、再度”センタサブソフトウェア要求信号”をセンタ１に送りダウンロードを行ない、ソフトウェアを使用する（Ｓ１０５，Ｓ１０６，Ｓ１０７，Ｓ１０８）。

【００７０】ソフトウェアの使用中に必要なくなったセンタサブソフトウェアは使用中に随時クリアされる。また、ダウンロードされたセンタサブソフトウェアで、ソフトウェア使用終了後に消去されていなかったものは全てクリアされる（Ｓ１０９）。そしてセンタサブソフトウェアが消去された欠落パートは初期化される。

【００７１】

【発明の効果】本発明によれば、アプリケーションソフトウェアは端末とセンタの両方に分割されて保持され、端末上のソフトウェアは単独で動作することができなため、ソフトウェアが動作していない時に、端末において解析しただけでは、アプリケーションソフトウェア全体の構成を知ることは不可能である。

【００７２】また端末利用者がアプリケーションソフトウェアを動作させる時には必ずセンタと回線を利用して接続し、センタ上のソフトウェアをダウンロードして端末上のソフトウェアと結合する必要があるとともに、ダウンロードされるソフトウェアは毎回異なるように設定

され、結合された結果のソフトウェア配置も毎回異なり、さらにメモリ上の使用済みの部分は適宜消去されるため、ソフトウェア解析を困難なものとしている。

【００７３】このためアプリケーションソフトウェアの構造自体を複雑にすることなく、また暗号化することなく、アプリケーションソフトウェアの解析を困難にし、端末におけるアプリケーションソフトウェアの不正利用を防止することができる。

【図面の簡単な説明】

【図１】本発明の一実施例に係わるソフトウェア解析保護方法を実施するシステムの構成を示すブロック図である。

【図２】上記実施例において端末サブソフトウェアとセンタサブソフトウェアの関係を示す模式図である。

【図３】上記実施例において、センタサブソフトウェア＊をクリアする方法の一例を示す模式図である。

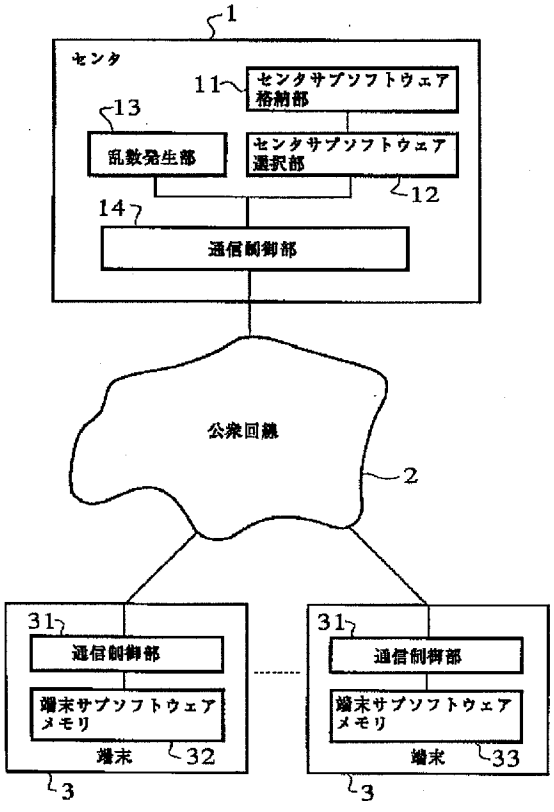
【図４】上記実施例において、センタサブソフトウェアをクリアする方法の他の例を示す模式図である。

【図５】上記実施例におけるシステム全体の動作例を示すシーケンスチャートである。

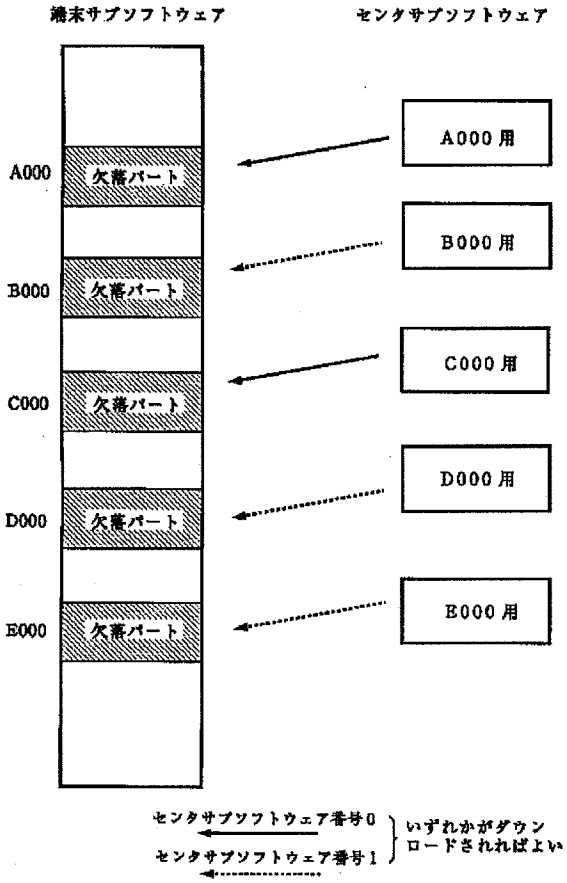
【符号の説明】

１　センタ
２　公衆回線
３　端末
１１　センタサブソフトウェア格納部
１２　センタサブソフトウェア選択部
１３　乱数発生部
１４，３１　通信制御部
３２　端末サブソフトウェアメモリ

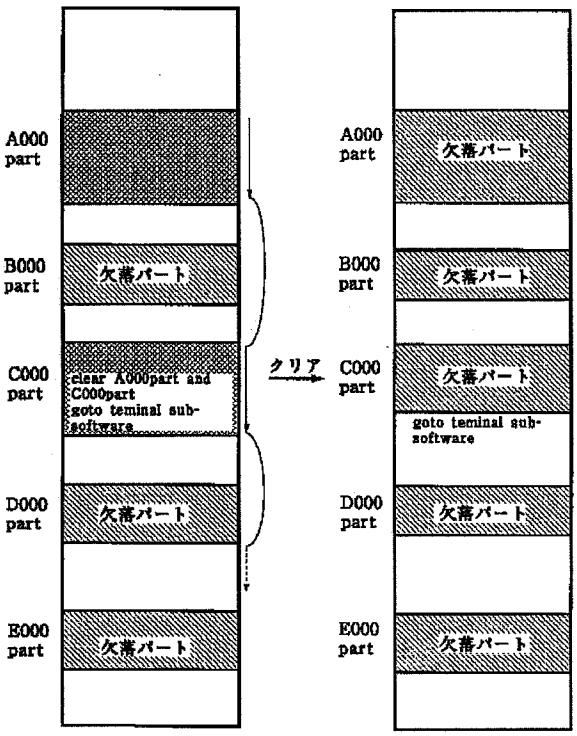
【図１】



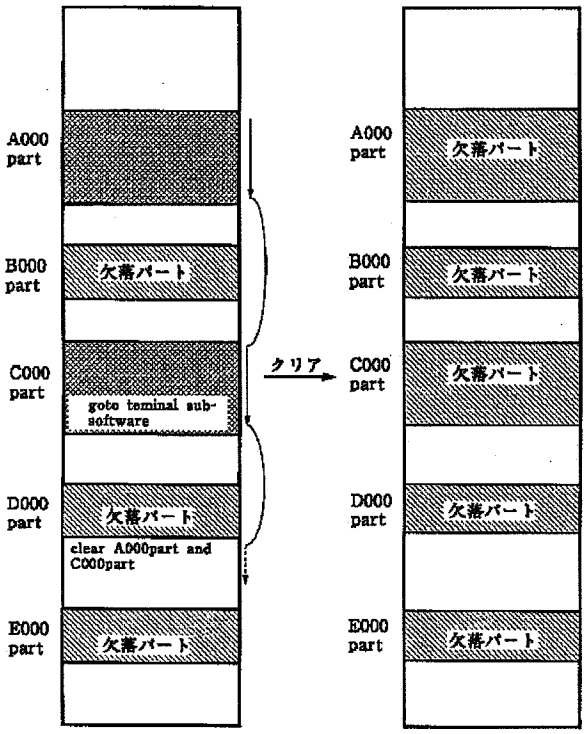
【図２】



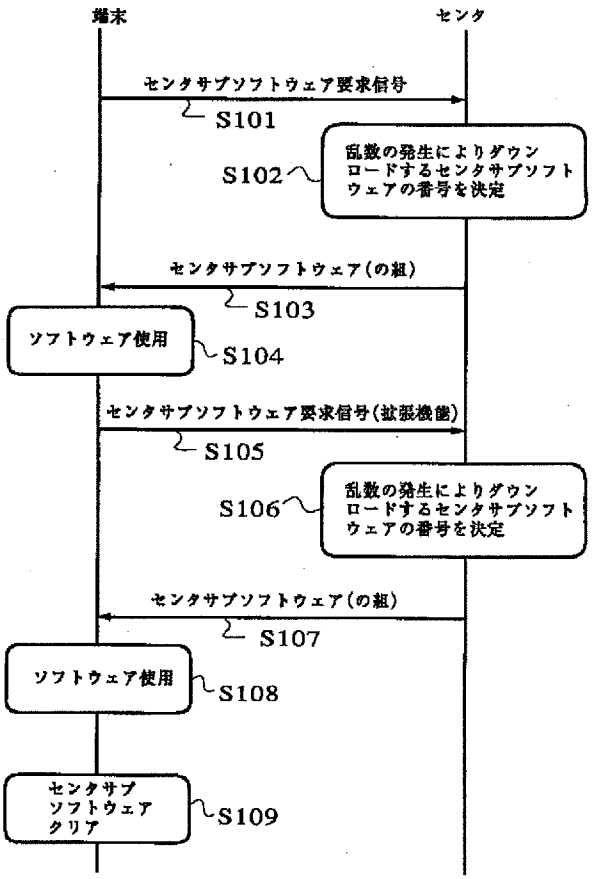
【図３】


A000 part
B000 part
C000 part
D000 part
E000 part
欠落パート
clear A000part and C000part goto teminal sub-software
クリア
goto teminal sub-software

【図４】


A000 part
B000 part
C000 part
D000 part
E000 part
欠落パート
goto teminal sub-software
クリア
clear A000part and C000part

【図５】


端末
センタ
センタサブソフトウェア要求信号
S101
S102
乱数の発生によりダウンロードするセンタサブソフトウェアの番号を決定
センタサブソフトウェア(の組)
S103
ソフトウェア使用
S104
センタサブソフトウェア要求信号(拡張機能)
S105
S106
乱数の発生によりダウンロードするセンタサブソフトウェアの番号を決定
センタサブソフトウェア(の組)
S107
ソフトウェア使用
S108
センタサブソフトウェアクリア
S109

フロントページの続き

(72)発明者　三宅　延久
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内

(72)発明者　寺内　敦
東京都千代田区内幸町１丁目１番６号　日本電信電話株式会社内